지능로봇에서 에이전트와 ESB를 사용한 서비스 지향 애플리케이션의 자가 재구성

(Self-Reconfiguration of Service-Oriented Application using Agent and ESB in Intelligent Robot)

이 재 정 † 김 진 한 †
(Jaejeong Lee) (Jinhan Kim)
이 창 호 †† 이 병 정 †††
(Changho Lee) (Byungjeong Lee)

요 약 지능로봇(Intelligent Robot)은 주변환경을 감지하는 센서로부터 실시간 정보를 수집하고 지능적인 기능을 수행한다. 지능로봇의 자가 재구성(Self-Reconfiguration) 능력은 외부 환경의 변화에 대응하기 위해 기능을 재구성하고, 오류가 발생하였을 때 중지 없이 스스로 회복할 수 있는 중요한 요소이다. 본 논문에서는 ESB(Enterprise Service Bus)를 사용한 지능로봇의 에이전트 기반 자가 재구성 프레임워크를 제안한다. 본 논문의 프레임워크는 멀티 에이전트 시스템을 이용한 서비스 지향 애플리케이션의 동적인 발견과 자가 재구성에 초점을 맞춘다. 지능로봇이 예외적인 상황을 만났을 때, 지능로봇은 외부의 서비스 저장소로부터 새로운 서비스를 다운로드후 실행시켜 상황을 해결한다. 에이전트 기술은 로봇들이 상호작용하기 위한 지능적인 접근법을 제공하고, ESB는 분산된 서비스 또는 지식을 활용하고 조직하기 위한 방법을 제공한다. 또한 본 연구의 유효성을 보여주기 위해 프로토타입을 구현하였다.

키워드 : 자가 재구성, 지능로봇, 에이전트, ESB

Abstract Intelligent Robots (IR) get data of the current situation from sensors and perform knowledgeable services. Self-reconfiguration of IR is an important factor to change itself without stopping while supporting environment and technology change. In this paper, we propose an agent based self-reconfiguration framework of IR using ESB (Enterprise Service Bus). This framework focuses on dynamic discovery and reconfiguration of service-oriented applications using multi-agent system in intelligent robots. When IR meets an irresolvable situation it downloads a necessary service agent from an external service repository, executes the agent, and resolves the situation. Agent technology provides an intelligent approach for collaborations of IR. The prototype has also been implemented to show the validity of our study.

Key words : Self-Reconfiguration, Intelligent Robot, Agent, ESB

1. 서 론

지능로봇은 항상 사용자와 상호작용하고, 현재의 주변환경을 이해하고, 사용자에게 필요한 서비스를 제공하는 네트워크와 소프트웨어 기반의 로봇이다. 지능로봇은 주변환경을 감지하는 센서로부터 실시간 정보를 표현하고 특수한 목적에 맞추어진 적응을 위한 정보와 서비스를 제공한다.

지능로봇은 예측하기 어려운 환경에서도 새로운 서비스를 발견하여 적응하기 위하여 스스로를 재구성할 수 있어야 한다. 이러한 소프트웨어 적응 문제를 해결하기 위하여 SOC(Service-Oriented Computing)[1]는 이종의 분산된 서비스 또는 지식을 활용하고 조직하기 위한 주요한 방법을 제공한다.

지능로봇의 자가 재구성을 위해, 먼저 의미적(semantic) 정보[2,3]를 사용해서 알맞은 서비스를 검색한다. 그리고 지능로봇은 중지하지 않고 적합한 서비스를 사용해서 동적으로 서비스를 재구성해야 한다[4]. 그러나 이전 연구들은 컴포넌트 소프트웨어의 생명주기를 효율적으로 관리하지 못하는 한계를 가지고 있다. 또한 서비스 기능 개념(concept)들 간의 매칭 정도(degree of matching)를 포함한 상세한 의미적 정보를 사용하여 검색을 수행하지 못하였다.

· 본 연구는 21세기 프론티어기술개발사업인 인간기능생활지원지능로봇 기술개발사업단의 기술비 지원으로 수행되었음
· 이 논문은 제34회 추계학술대회에서 '지능로봇에서 에이전트와 ESB를 사용한 서비스 지향 애플리케이션의 자가 재구성'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 서울시립대학교 컴퓨터과학부
jaejeong2@gmail.com
kimjinhan@gmail.com
†† 비 회 원 : WOOJIN inc 계측기술 연구소
chlee@woojininc.com
††† 종신회원 : 서울시립대학교 컴퓨터과학부 교수
bjlee@uos.ac.kr

논문접수 : 2007년 12월 7일
심사완료 : 2008년 9월 1일

정보과학회논문지: 컴퓨팅의 실제 및 레터 제14권 제8호(2008.11)

본 논문에서는 ESB를 사용한 지능로봇의 에이전트 기반 자가 재구성 프레임워크를 제안한다. 이 프레임워크는 멀티에이전트 시스템을 이용한 서비스 지향 애플리케이션의 동적인 발견과 자가 재구성을 지원한다. 자율성, 협동성, 상호관계적인 에이전트의 특징들은 자가 재구성 애플리케이션의 지능적인 접근법을 위해 사용되었다. 그리고 ESB는 웹 서비스와 기존의 레거시 시스템(legacy system) 등을 포함한 다양한 서비스 또는 자원들의 생명주기를 관리하는 효율적인 방법을 제공한다.

본 논문의 구성은 다음과 같다. 2장에서 자가 재구성과 시맨틱 웹 서비스에 관한 연구를 기술한다. 3장에서는 서비스 발견 및 자가 재구성 프레임워크를 기술한다. 4장에서는 본 논문에서 제안한 프레임워크의 프로토타입을 설명한다. 마지막으로 5장에서 향후 연구를 기술하고 결론을 맺는다.

2. 관련 연구

소프트웨어의 자가 재구성은 예외적인 상황을 다루는 능력이다. 자가 재구성은 사용자 요구사항이나 환경의 변화에 대응하기 위한 소프트웨어의 필수적인 능력이다[5]. 자가 재구성 능력을 가진 시스템은 환경의 변화에 대응하기 위해 스스로 기능을 재구성하고, 오류가 발생하였을 때 회복할 수 있고, 외부의 영향으로부터 방어할 수 있어야 한다. 이러한 목적을 달성하기 위해서 시스템은 스스로 내부의 상태를 인식하고, 시스템 외부 환경의 변화를 모니터링하고, 스스로 적절하게 재구성할 수 있어야 한다. 이를 위한 아키텍처는 일반적으로 모니터링, 결정, 재구성의 세가지 부분으로 나누어진다. 모니터링 부분은 센서뿐만 아니라 시스템 내부 상태를 통해 환경의 변화를 감지한다. 모니터링 정보는 결정 부분으로 보내져서 시스템이 어떻게 변화를 다룰지를 판단한다. 결정 부분의 판단 결과에 따라서 재구성 부분에서 시스템을 재구성한다[6].

기존 연구에서는 사용자의 요구사항 변화에 필요한 새로운 서비스를 의미정보를 이용하여 발견하는 기법이 연구되었다[4]. 웹 서비스를 사용하는 것은 동적으로 새로운 서비스를 조합할 수 있다는 것을 의미한다. 이를 위해 [7]의 연구에서 ESB 기반의 시스템의 동적인 자가 재구성이 제안되었다. 그러나 이 연구는 시스템의 재구성에 초점을 맞추고 있고 환경을 모니터링하고 결정을 내리는 기능은 불충분하다. 본 논문의 결정관리자는 상황정보를 이용하여 로봇의 주변의 상황정보를 모델링하고 분석을 통해서 재구성을 위한 결정을 내린다.

3. 지능로봇의 자가 재구성 프레임워크

3.1 ESB 기반 에이전트 아키텍처

본 논문에서 자가 재구성 프레임워크는 서비스 재구성 계층과 서비스 발견 계층으로 나뉜다(그림 1). 먼저 로봇내부 서비스 재구성 계층의 결정관리자(decision manager)는 센서 혹은 사용자로부터 재구성 이벤트를 받았을 때 현재 상황을 판단하여 재구성을 위한 결정을 내린다. 그리고 재구성관리자(reconfiguration manager)는 결정관리자의 명령에 따라 재구성을 수행하고 JBI (Java Business Integration)[8]서비스들의 생명주기를 관리한다. JBI 서비스는 레거시 모듈, 웹 서비스 그리고 소프트웨어 컴포넌트와 같은 자원들을 포함한 BPEL (Business Process Execution Language)에 표현된 서비스 조합이다.

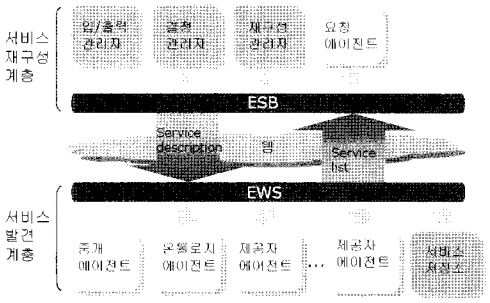

그림 1 실행시간 발견 및 자가 재구성 프레임워크

프레임워크의 서비스 발견 계층을 구성하는 로봇 외부의 에이전트들은 서비스 저장소(service repository)에 서비스를 등록하고 검색하기 위해 서로 상호작용한다.

요청 에이전트(request agents)는 서비스 설명 (service description)을 사용하여 서비스를 요청하고 로봇 시스템으로 결과를 제공하는 인터페이스 에이전트이다. 요청 에이전트는 중개 에이전트(broker agent)로 서비스 검색 요청을 전달한다. 서비스 요청 설명은 적합한 서비스를 발견하는데 필요한 정보를 포함하고 있다. 이러한 정보는 시스템의 요구사항을 표현하기 위한 개념(concept)과 IOPE(Input, Output, Precondition, Effect)[9]같은 서비스의 기능적 정보를 포함한다. 중개 에이전트는 요청 에이전트로부터 서비스 검색 요청을 받고 요청 설명을 이용해서 서비스 저장소에서 적합한 서비스를 검색한다. 제공자 에이전트(provider agent)는 서비스 제공자로부터 서비스를 받아 서비스의 기능적인 정보를 함께 서비스 저장소에 등록한다.

3.2 에이전트 기반 서비스 검색

에이전트는 사용자 대신에 어떤 작업을 수행하는 자율적인 프로세스이고 독립 또는 어떤 환경의 일부분으로 행동한다. 그림 2는 프레임워크에서 동적인 서비스 검색과 매칭의 절차를 보여준다.

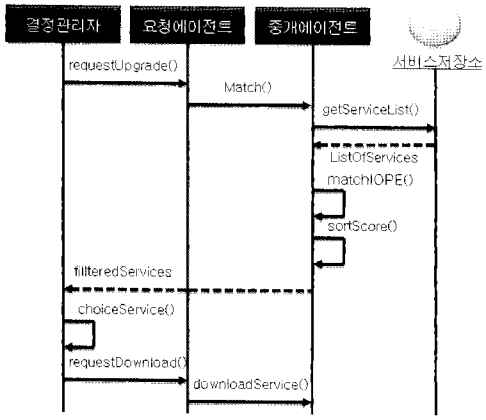

그림 2 서비스의 매칭 및 랭킹 절차

먼저 결정관리자는 요청 에이전트에게 서비스 업그레이드를 위한 요청 메시지를 보낸다. 요청 에이전트가 중개 에이전트에게 서비스 검색 메시지를 전달하면, 중개 에이전트는 서비스 저장소로부터 모든 서비스의 목록을 가져온다. 중개 에이전트는 서비스가 적합한지 아닌지를 판단하기 위해 서비스 IOPE와 요청 IOPE를 비교한다. 중개 에이전트는 요청설명의 가중치 값(weight value)을 이용해서 매칭 정도에 따라 적합한 서비스의 점수(score)를 계산한다[10]. 그리고 계산된 점수 값에 따라서 적합한 서비스를 내림차순으로 정렬한다. 그리고 중개 에이전트는 결정관리자에게 적합한 서비스의 목록을 되돌려준다.

결정관리자는 가장 적합한 서비스 하나를 선택하고 해당하는 서비스 다운로드를 결정한다. 결정관리자는 요청 에이전트에게 해당 서비스의 컴포넌트 파일을 지정된 폴더에 다운로드할 것을 요청한다. 요청 에이전트는 중개 에이전트를 통해서 서비스 저장소로부터 선택된 서비스 컴포넌트를 얻는다. 중개 에이전트는 SAAJ (SOAP with Attachments API for Java)-XML 문서를 전송하는 표준방법을 이용해서 요청 에이전트에게 서비스 컴포넌트를 전송한다.

```
<service>
<name>InfraredApp</name>
<hasInput>(direction ?X)</hasInput>
<hasOutput>(image ?Y)</hasOutput>
<hasPrecondition>(lux 20)</hasPrecondition>
<hasEffect>(validImage ?Y)</hasEffect>
<textDescription>
(weight(I=1,O=1,P=1,E=1))
</textDescription>
</service>
```

그림 3 서비스 검색을 위한 요청 설명

그림 3은 서비스 검색을 위한 서비스 요청 설명을 보여준다. hasInput, hasOutput, hasPrecondition, 그리고 hasResult 요소는 서비스의 IOPE를 표현한다. 각 요소의 가중치 값은 textDescription 요소에 표현하며, 요청자의 정확한 환경의 제약을 나타내기 위하여 이러한 가중치 값을 사용한다. 그림 2의 matchIOPE 함수에서 각 IOPE 요소의 점수를 계산하기 위해 IOPE 가중치를 사용한다[10].

3.3 소프트웨어 자가 재구성

소프트웨어 자가 재구성은 시스템을 중지하는 일없이 환경 변화를 감지하고 재구성할 수 있는 메커니즘 이다. 본 프레임워크의 센서 모듈은 주변 환경으로부터 상황 변화 정보를 감지하고, 결정관리자는 센서로부터 전달되는 정보를 이용해서 재구성을 위한 결정을 내린다. 결정관리자의 역할은 주변 환경 변화 정보를 모니터링하고 감지하고 현재 로봇을 재구성 할 필요가 있을 때 재구성을 위한 계획을 수립하는 것이다. 결정관리자는 이를 위해 몇 가지 정보를 분석한다. 먼저 환경 변화의 정보를 활용하는 규칙(rule), 상황정보(context), 목표정보(goal)를 가진다. 결정관리자는 이러한 정보를 이용해서 현재의 상황을 분석하고 수정한다. 규칙들은 상황정보의 변화로부터 동적인 재구성 수행 여부를 결정할 수 있는 정보이다. 결정관리자는 규칙들을 통해 얻은 결정과 이 로봇의 목표정보를 토대로 새로운 계획을 수립한다. 계획 수립 절차는 결정관리자 내부의 기능 온톨로지(function ontology)와 서비스 온톨로지(service ontology)를 가지고 수행된다[3]. 먼저 기능 온톨로지의 개념이 새로운 기능을 발견하기 위해 검색된다. 다음에 검색된 기능을 기반으로 서비스 온톨로지의 서비스들을 개념 매칭을 통해 검색한다. 그리고 서비스를 재구성하기 위해 재구성관리자에게 전달되는 명령의 집합이 생성된다. 재구성관리자는 현재 시스템의 서비스 저장소에 저장된 서비스의 조합 정보를 검사한다. 만약 저장소가 재구성을 위해 요구되는 적합한 서비스를 가지고 있다면 재구성관리자는 ESB를 통해서 서비스의 배포, 실행, 중지와 같은 명령을 수행한다. 만약 저장소에 적합한 JBI 서비스가 없다면 재구성관리자는 요청 에이전트에게 새로운 서비스 검색을 요청한다. 재구성관리자는 각 서비스의 생명주기와 종속성 정보를 가지고 있기 때문에 재구성관리자는 각 서비스의 상태를 모니터링할 수 있다.

4. 지능로봇 시뮬레이터

본 연구에서는 ESB를 사용한 지능로봇의 에이전트 기반 자가 재구성을 보여주기 위해 로봇 시뮬레이터와 EWS(Extended Web Service)를 구현하였다. 서비스들은 에이전트들과 UDDI(Universal Description, Disco-

very, and Integration) 그리고 서비스 저장소로 구성된 EWS에 저장된다.

본 논문에서는 에이전트 구현을 위하여 JADE(Java Agent Development Framework)를 사용하였다[11]. 로봇 시뮬레이터는 로봇이 업무를 수행하는 동안 예기치 않은 상황에 직면했을 때 EWS의 새로운 서비스를 요청한다. 다음은 로봇 시뮬레이터를 위한 시나리오를 보여준다.

- 서비스 제공자는 서비스와 속성정보를 EWS에 등록한다.
- 불이 꺼지거나 켜지는 환경 변화를 발생시킨다(예: 로봇이 존재하는 방 전등이 켜짐/꺼짐).
- 로봇은 EWS의 서비스 명세를 사용해서 새로운 서비스를 요청한다.
- EWS는 요청 설명을 사용해서 UDDI와 서비스 저장소를 검색한다. 그리고 적합한 서비스의 목록을 반환한다.
- 로봇은 리스트로부터 가장 적합한 서비스를 선택하고 다운로드를 요청한다.

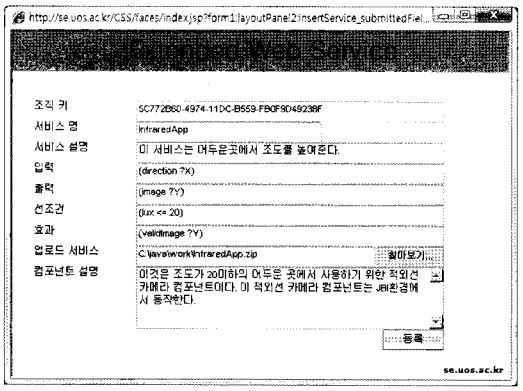

그림 4 서비스 등록화면

그림 4는 서비스와 속성정보를 EWS에 등록하는 화면이다. 속성정보는 UDDI 등록을 위한 조직 키(organization key) 값, 서비스 이름, 서비스의 기능을 표현하기 위한 IOPE 정보를 포함한다. 그림 4에서 적외선 서비스의 입력과 출력은 방향과 이미지이다. 선조건은 밝기 정도가 낮은 어두운 상태를 나타내고, 효과는 출력 이미지가 유효한지를 나타낸다. 서비스 제공자는 제공자 에이전트에게 해당 서비스의 업로드 요청을 보낸다. 제공자 에이전트는 중개 에이전트에게 서비스 속성정보와 서비스를 EWS에 등록 요청한다.

그림 5는 로봇내부에서 수행되는 요청 에이전트와 EWS에서 수행되는 중개 에이전트를 보여준다. 그림 6은 로봇이 방안에 동작하는 모습을 보여주는 로봇 시뮬

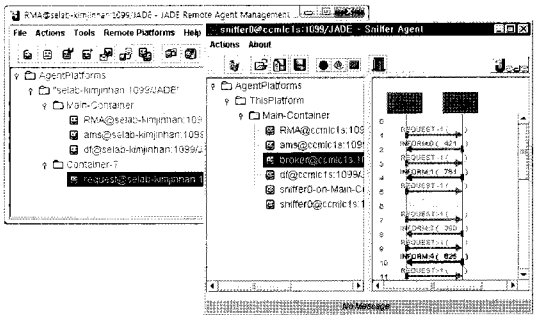

그림 5 EWS와 로봇에서 수행중인 에이전트

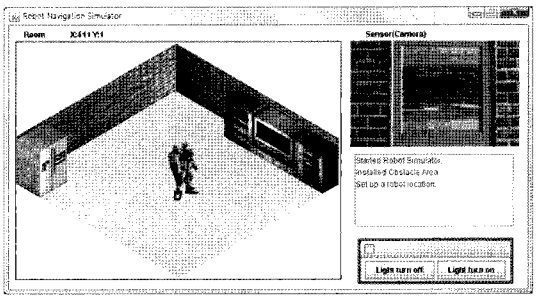

그림 6 자가 재구성 수행전의 로봇 시뮬레이터

레이터를 보여준다. 화면 왼쪽은 방안의 로봇의 행동이 나타나고, 오른쪽은 카메라의 화면, 로봇의 정보를 보여주는 메시지 창 그리고 방안의 조명을 조절할 수 있는 버튼을 포함한다. 그림 6에서 방안의 조명은 밝게 켜져 있다. 메시지 창은 로봇이 현재 비전 카메라 서비스를 사용한다고 나타낸다.

그림 7은 그림 6의 상황정보를 모델링한 모습을 보여주고 있다. 전체 환경은 robot과 room으로 구성되고 room은 door, light 그리고 wall로 구성된다. 센서로부터 전달받은 정보는 이 상황정보의 변화이다. 이러한 변화는 규칙을 적용하여 현재 상황을 판단한다.

그림 8은 로봇에서 사용되는 규칙들의 일부를 Java 기반 규칙 엔진 Jess(Java Expert System Shell) 코드로 보여주고 있다. 그림 7에서 정의한 상황모델 요소를 사용하여 규칙을 정의하였다. 그림 8의 첫 번째 규칙은

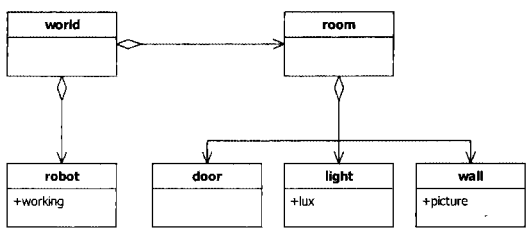

그림 7 그림 6의 로봇 상황정보(context)

```
(defrule infrared
    (and (robot (working on)) (light (lux <= 20)))
  => (assert (result (value infrared))))
(defrule vision
    (and (robot (working on)) (light (lux >  20)))
  => (assert (result (value vision))))
```

그림 8 로봇 규칙 예

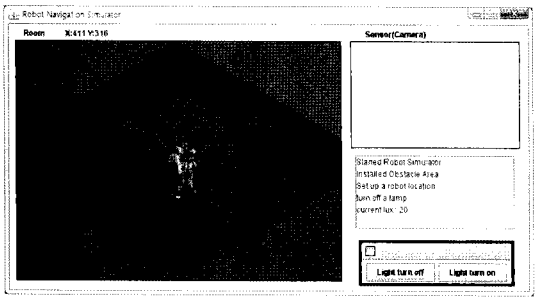

그림 9 불이 꺼진 상태의 로봇 시뮬레이터

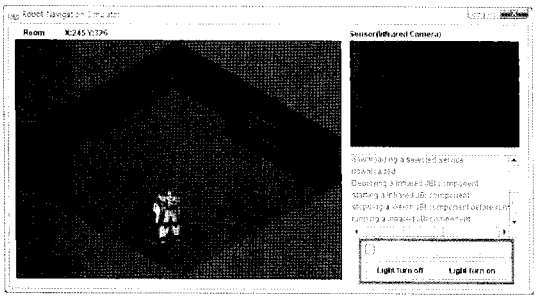

그림 10 자가 재구성 수행후의 로봇 시뮬레이터

로봇이 동작 중이고 조명의 밝기가 20 lux 이하 일 때, 적외선 카메라 모듈을 이용해야 된다는 것을 기술하고 있다. 반대로 두 번째 규칙은 비전 카메라 모듈을 이용해야 된다는 것을 기술하고 있다.

그림 9는 로봇이 업무를 수행하는 도중에 방안의 조명이 어두워지는 상태를 보여준다. 조명이 꺼지면 로봇은 카메라를 통해 아무것도 볼 수가 없다. 그림 9의 메시지 창은 방안의 조도(lux)가 20으로 낮아진 상태를 보여주고 있다. 이때 결정관리자는 상황정보 변화를 입력받아 규칙을 적용하여 현재 상황을 판단하고 기능 온톨로지와 서비스 온톨로지를 통해 필요로 하는 컴포넌트를 재구성관리자에게 요청한다.

그림 10은 실행시간에 새로운 서비스를 동적으로 다운로드하고 배포하고 그리고 서비스를 시작한 자가 재구성 후의 화면을 보여준다. 그림 10의 카메라 화면은 로봇이 실행시간 중에 적외선 카메라 서비스를 적용해서 어둡지만 방안의 모습을 보여주고 있다. 또한 메시지 창은 현재 재구성된 적외선 서비스가 배포된 상태를 보여준다.

5. 결론 및 향후 연구

본 논문에서는 ESB를 사용한 지능로봇의 에이전트 기반 자가 재구성 프레임워크를 제안하였다. 하지만 본 프레임워크는 지능로봇과 EWS 저장소간의 네트워크가 인터넷을 통해서 연결되기 때문에 실행 시에 성능의 제한을 가진다. 향후의 연구에서 로봇이 특수한 환경에 적응할 때 지능로봇의 성능을 개선하기 위한 연구가 수행되어야 할 것이다. 또한 상황정보를 가지고 추론(reasoning)을 통해서 로봇의 행동을 예측함으로써 로봇의 성능을 개선할 필요가 있다. 이러한 목적을 달성하기 위해 앞으로 체계적으로 로봇의 상황정보를 분석하고 로봇의 행동을 결정하는 연구가 필요하다.

참 고 문 헌

[1] OASIS, Reference Model for Service Oriented Architecture 1.0, Feb. 2006.

[2] A. Mikroyannidis, "Toward a Social Semantic Web," IEEE Computer, Vol.40, No.11, pp. 113-115, Nov. 2007.

[3] M. Uschold and M. Gruninger, "Ontologies: Principles, Methods and Applications," Knowledge Engineering Review, Vol.11, No.2, pp. 93-136, 1996.

[4] S. Chaiyakul, K. Limapichat, A. Dixit and E. Nantajeewarawat, "A Framework for Semantic Web Service Discovery and Planning," Proc. of IEEE Conference on Cybernetics and Intelligent Systems, pp. 1-5, June 2006.

[5] H. Koo and I. Ko, "A repository framework for self-growing robot software," Proc. of Asia-Pacific Software Engineering Conference, 2005.

[6] M. Hinchey and R. Sterritt, "Self-Managing Software," IEEE Computer, Vol.39, Iss. 2, pp. 107-109, Feb. 2006.

[7] S. Chang, J. Bae, W. Jeon, H. La and S. Kim, "A Practical Framework for Dynamic Composition on Enterprise Service Bus," Proc. of IEEE International Conference on Services Computing, 2007.

[8] J. ji-chen and G. Ming, "Enterprise service Bus and an Open Source Implementation," Proc. Of International Conference on Management Science and engineering, pp. 926-930, 2006.

[9] W3C, OWL-S: Semantic Markup for Web Services, Nov. 2004.

[10] J. Kim, J. Lee, B. Lee, "Runtime Service Discovery and Reconfiguration using OWL-S based Semantic Web Service," Proc. of IEEE 7th International Conference on Computer and Information Technology, Nov. 2007.

[11] F. Bellifemine, A. Poggi and G. Rimassa, "JADE, A FIPA2000 Compliant Agent Development Environment," AGENTS'01, May 2001.